

Kolmogorov structure functions for automatic complexity in computational statistics

Bjørn Kjos-Hanssen**

University of Hawai'i at Mānoa, Honolulu, HI 96822, USA,
 bjoernkh@hawaii.edu, <http://math.hawaii.edu/wordpress/bjoern/>

Abstract. For a finite word w of length n and a class of finite automata \mathcal{A} , we study the Kolmogorov structure function h_w for automatic complexity restricted to \mathcal{A} . We propose an approach to computational statistics based on the minimum p -value of $h_w(m)$ over $0 \leq m \leq n$. When \mathcal{A} is the class of all finite automata we give some upper bounds for h_w . When \mathcal{A} consists of automata that detect several success runs in w , we give efficient algorithms to compute h_w . When \mathcal{A} consists of automata that detect one success run, we moreover give an efficient algorithm to compute the p -values.

1 Introduction

Shallit and Wang [6] introduced automatic complexity as a computable alternative to Kolmogorov complexity. They considered deterministic automata, whereas Hyde and Kjos-Hanssen [4] studied the nondeterministic case, which in some ways behaves better. Unfortunately, even nondeterministic automatic complexity is somewhat inadequate. The string 00010000 has maximal nondeterministic complexity, even though intuitively it is quite simple. One way to remedy this situation is to consider a structure function analogous to that for Kolmogorov complexity.

The latter was introduced by Kolmogorov at a 1973 meeting in Tallinn and studied by Vereshchagin and Vitányi [8] and Staiger [7].

The Kolmogorov complexity of a finite word w is roughly speaking the length of the shortest description w^* of w in a fixed formal language. The description w^* can be thought of as an optimally compressed version of w . Motivated by the non-computability of Kolmogorov complexity, Shallit and Wang studied a deterministic finite automaton analogue.

Definition 1 (Shallit and Wang [6]). *The automatic complexity of a finite binary string $x = x_1 \dots x_n$ is the least number $A_D(x)$ of states of a deterministic finite automaton M such that x is the only string of length n in the language accepted by M .*

** This work was partially supported by a grant from the Simons Foundation (#315188 to Bjørn Kjos-Hanssen). The author also acknowledges the support of the Institute for Mathematical Sciences of the National University of Singapore during the workshop on *Algorithmic Randomness*, June 2–30, 2014.

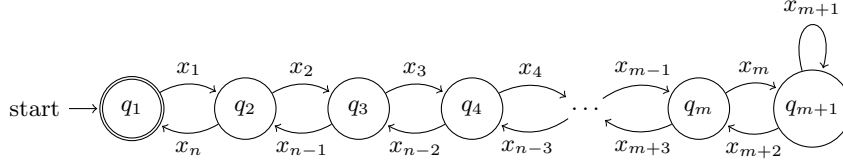


Fig. 1: A nondeterministic finite automaton that only accepts one string $x = x_1x_2x_3x_4 \dots x_n$ of length $n = 2m + 1$.

Hyde and Kjos-Hanssen [4] defined a nondeterministic analogue:

Definition 2. *The nondeterministic automatic complexity $A_N(w)$ of a word w is the minimum number of states of an NFA M , having no ϵ -transitions, accepting w such that there is only one accepting path in M of length $|w|$.*

The minimum complexity $A_N(w) = 1$ is only achieved by words of the form a^n where a is a single letter.

Definition 3. *Let $n = 2m + 1$ be an odd number. A finite automaton of the form given in Figure 1 for some choice of symbols x_1, \dots, x_n and states q_1, \dots, q_{m+1} is called a Kayleigh graph¹.*

Theorem 4 (Hyde [3]). *The nondeterministic automatic complexity $A_N(x)$ of a string x of length n satisfies*

$$A_N(x) \leq b(n) := \lfloor n/2 \rfloor + 1.$$

Proof. If the length of n is odd, then a Kayleigh graph witnesses this inequality. If the length of n is even, a slight modification suffices, see [3]. \square

Definition 5. *The complexity deficiency of a word x of length n is*

$$D_n(x) = D(x) = b(n) - A_N(x).$$

The structure function of a string x is defined by $h_x(m) = \min\{k : \text{there is a } k\text{-state NFA } M \text{ which accepts at most } 2^m \text{ strings of length } |x| \text{ including } x\}$. In more detail:

Definition 6 (Vereshchagin, personal communication, 2014, inspired by [8]). *In an alphabet Σ containing b symbols,*

$$h_x(m) = \min\{k : \exists k\text{-state NFA } M, x \in L(M) \cap \Sigma^n, |L(M) \cap \Sigma^n| \leq b^m\}.$$

We also define the “converse” structure function

$$g_x(m) = h_x(|x| - m)$$

and its maximum

$$G_n(m) = \sup_{|x|=n} g_x(m).$$

¹ The terminology is a nod to the more famous Cayley graphs as well as to Kayleigh Hyde’s first name.

2 Basic properties

Generalizing Hyde's result [4] that $h_x(0) \leq h_{xy}(0)$, we have

Theorem 7.

$$h_x(m) \leq h_{xy}(m)$$

and hence for $b \in \{0, 1\}$,

$$h_x(|x| - k) = h_x(n - k) \leq h_{xb}(n - k) = h_{xb}(|xb| - (k + 1))$$

$$g_x(k) \leq g_{xb}(k + 1)$$

hence

$$G_n(k) \leq G_{n+1}(k + 1)$$

Conjecture 8. $G_n(k) \leq G_{n+1}(k)$.

Conjecture 9 (The right upper bound conjecture). $\forall k \lim_n G_n(k) = k + 1$.

We have verified Conjecture 9 for $k = 0$ and $k = 1$ by simple proofs.

Theorem 10. *There exist x and b with $h_x(|x| - 1) \not\leq h_{xb}(|xb| - 1)$.*

Proof. Let $x = 0100$ and $b = 0$. Then $h_x = (3, 3, 2, 2, 1)$ and $h_{xb} = (3, 3, 2, 2, 2, 1)$. That is, the string 01000 is accepted by an automaton with 2 states accepting only 1/16th of all strings of length 5, but 0100 cannot be accepted by any such automaton for length 4. \square

Theorem 11. *We have the following inequalities for all strings x and y of length n , symbols b , and $0 \leq m \leq n$.*

$$h_x(|x| - 1) + 1 \geq h_{xb}(|xb| - 2)$$

$$h_x(m) + |y| \geq h_{xy}(m)$$

$$g_x(n - m) + |y| \geq g_{xy}(n + |y| - m)$$

$$g_x(k) \leq g_{xb}(k + 1) \leq g_x(k) + 1$$

Proof. Suppose there is M accepting only half the strings of length n , including x . Then there is M' accepting only a quarter of the strings of length $n + 1$, including xb , with one extra state (namely, just add a new state and one new edge labeled by b). \square

Definition 12. *The entropy function $\mathcal{H} : [0, 1] \rightarrow [0, 1]$ is given by*

$$\mathcal{H}(p) = -p \log_2 p - (1 - p) \log_2 (1 - p).$$

It is then fairly canonical to define the inverse entropy function $\mathcal{H}^{-1} : [0, 1] \rightarrow [0, 1/2]$ by

$$\mathcal{H}^{-1}(y) = x \iff \mathcal{H}(x) = y \quad \text{and} \quad 0 \leq x \leq 1/2.$$

Theorem 13. For $0 \leq k \leq n$,

$$\log_2 \binom{n}{k} = \mathcal{H}(k/n)n + O(\log n).$$

Proof. Let $\log = \ln = \log_e$. For $u \in \mathbb{N}$, let

$$S_u = \sum_{k=2}^u \log k, \quad I_u = \int_1^u \log x \, dx, \quad \text{and} \quad J_u = \int_2^{u+1} \log x \, dx.$$

Let

$$\alpha_n = \log \binom{n}{k} = S_n - S_k - S_{n-k}.$$

Note $I_u \leq S_u \leq J_u$ and

$$J_u - I_u = \int_u^{u+1} \log x \, dx - \int_1^2 \log x \, dx \leq \log(u+1),$$

Thus up to $O(\log n)$ error terms we have

$$\begin{aligned} \alpha_n &= \int_1^n \log x \, dx - \int_1^k \log x \, dx - \int_1^{n-k} \log x \, dx \\ &= (n \log n - n) - (k \log k - k) - [n - k \log(n - k) - n - k] \\ &= n \log n - k \log k - n - k \log(n - k) \\ &= -k \log(k/n) - n - k \log(1 - k/n) \end{aligned}$$

and hence

$$\log_2 \binom{n}{k} = -k \log_2(k/n) - n - k \log_2(1 - k/n) = \mathcal{H}(k/n) \cdot n.$$

□

Theorem 14. Suppose the number of 0s in the binary string x is $p \cdot n$. Then

$$h_x(\mathcal{H}(p)n) \leq pn + O(\log n).$$

Proof. Consider an automaton M as in Figure 3 that has $[pn]$ many states, and that has one left-to-right arrow labeled 0 for each 0, and a loop in place labeled 1 for each consecutive string of 1s. Since M accepts exactly those strings that have $[pn]$ many 0s, the number of strings accepted by M is $\binom{n}{[p \cdot n]}$. By Theorem 13 this is $\leq 2^k$ approximately when $\mathcal{H}(p)n \leq k$, and we are done. □

Example 15. A string of the form $0^a 1^{n-a}$ satisfies $h_x(\log_2 n) = 2$ whereas $h_x(0)$ may be $n/2$. For instance 0011 has $h_x(2) = 2$. On the other hand $h_x(1) = 3$ which is why this string is more complicated than 0110.

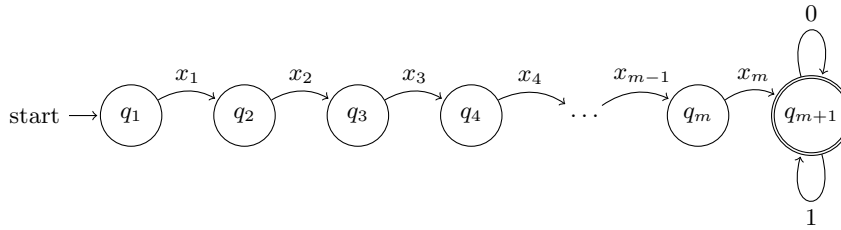


Fig. 2: An automaton illustrating the linear upper bound on the automatic structure function from Theorem 16.

Theorem 16. *For any x of length n ,*

$$1 \leq h_x(m) \leq n - m + 1 \text{ for } 0 \leq m \leq n.$$

Proof. $1 \leq h_x(n - k) \leq k + 1$ because we can start out with a sequence of determined moves, after which we accept everything, as in Figure 2. \square

Remark 17. *We do not know whether $h_w(0)$, i.e., $A_N(w)$, is polynomial-time computable as a function of w . However, $g_w(m)$ is polynomial-time computable for each fixed parameter m , since there is an upper bound that only depends on m by Theorem 16.*

Theorem 16 suffices to calculate h_x for $n = 0$. For x of length 0, we have $h_x(0) = 1$.

Theorem 18. *We have $h_x(0) = A_N(x)$, the automatic complexity of x . We have $h_x(m) \geq h_x(m + 1)$ for each $0 \leq m < n$.*

Theorem 19. *We have $h_x(n - k) \geq 2$ unless x is unary or $k = 0$.*

Proof. If x consists of both 0s and 1s then a 1-state automaton is useless. \square

3 An approach to computational statistics

We propose to study automatic sufficient statistics by looking at the p -value of an event $h_x(m) \leq q$. The m that gives the lowest p -value for a given x gives the model that we use to explain our data x . If this p -value is not less than a threshold such as Fisher's $p = 1/20$ then we just use a null hypothesis of an arbitrary binary string. Considering alphabets larger than size 2 is forced upon us in that the most complex string of length 4 is naturally 0123, and we thus get a better understanding of upper bounds. There is an automaton with 1 state accepting only binary strings. For a ternary alphabet we would redefine the structure function.

$$2^n \leq 3^m$$

when

$$n \log 2 \leq m \log 3$$

$$n \frac{\log 2}{\log 3} \leq m$$

Of course, $\log 2 / \log 3 = \log_3 2 = 0.63 \dots < 1$. Now the p -value for $h_x(\lceil n \log_3 2 \rceil) \leq 1$ should be $< 1/20$ as soon as n is sufficiently large, since the only thing you can do with one state is to limit the alphabet. Thus $h_x(m) \leq 1$, for $m < n$, only happens for $3 \cdot 2^n - 3$ out of the 3^n strings. For $n \geq 11$,

$$\frac{3 \cdot 2^n - 3}{3^n} = \frac{2^n - 1}{3^{n-1}} \leq \frac{1}{20}.$$

Thus a binary string of length 11 should lead to rejection of the null hypothesis that we have a random ternary string. If we observe a binary string and are considering the null hypothesis of a quaternary alphabet, we need

$$\frac{\binom{4}{2} 2^n - ((\binom{4}{2}) - 3) + 4}{4^n} = \frac{6 \cdot 2^n - 8}{4^n} = \frac{3 \cdot 2^{n-1} - 2}{4^{n-1}} < \frac{1}{20}$$

which gives $n - 1 \geq 6$, $n \geq 7$.

Of course, the probability of a binary $\{0, 1\}$ string in a 4-ary alphabet is just $(2/4)^n$, but here we account for possibilities $\{i, j\} \neq \{0, 1\}$.

Theorem 20. *Let x be a string of length n in an a -ary alphabet with uniform distribution.*

1. *The probability that x turns out to be binary, i.e., to be a string over some 2-element alphabet, is*

$$\frac{\binom{a}{2} 2^n - a(a-2)}{a^n}.$$

2. *The probability that x turns out to be binary over $\{0, 1\}$ is $(2/a)^n$.*
3. *The probability that x turns out to be $a-1$ -ary, i.e., to be a string over some $(a-1)$ -element alphabet, is*

$$a^{-n} \sum_{k=1}^{a-1} \binom{a}{k} (a-k)^n (-1)^{k+1}.$$

Proof. Part (2) is obvious and Part (3) follows from the inclusion-exclusion principle. We prove part (1). In the estimate $\binom{a}{a-1} (a-1)^n$, we are counting the a many unary strings a wrong number c_a of times, but how many times does not depend on n . To find the number c_a , we solve

$$\frac{\binom{a}{2} 2^1 - c_a}{a^1} = 1 = \frac{\binom{a}{2} 2^2 - c_a}{a^2}$$

which gives $c_a = a(a-2) = 2a(a-1) - a^2$. □

Corollary 21. *Let x be a string in an a -ary alphabet with uniform distribution. The probability that $h_x(m) = 1$ for some $m < n$ is 0 in the limit as $n \rightarrow \infty$.*

Example 22. *In the case $n = 6$, we need*

$$\frac{\binom{5}{2}2^n - 15}{5^n} = \frac{10 \cdot 2^n - 15}{5^n} < \frac{1}{20}$$

which is true for $n \geq 6$.

4 Run complexity

4.1 Algorithms for single-run complexity: p -values and structure function

One severe restriction that is certainly polynomial time computable is to require that each step should be from a state s to $s + 1$, except for one state that can have self-loops.

An implementation is available at [2].

Note that for a ternary alphabet the number of unary valences equals the number of binary valences. In general, we have to account for the number of valences when we assign a p -value. In a quaternary alphabet there are $\binom{4}{2}$ binary valences, but only 4 unary valences, which means that the p -values for an observed run has to be adjusted accordingly when compared with a run of a “different-ary” valence.

To specify an automaton we then only need to specify the location of the repeat, number of repeat cycles there, and labels on the edges. This is implicitly studied in Alikhani’s Master’s thesis [1]. In a ternary alphabet this could allow a block with a limited alphabet. This would give $n - \log_2 n$ as probabilistic upper bound on $h_x(0)$, and $h_x(m) \leq n - m + 1$.

If we have a 3-ary language with 2 self-loops at a given state, then the structure function goes up by $\lceil \log 3 / \log 2 \rceil = 2$ not 1 as we decrease m from that point. Actually it varies because it is $\lceil (m - mm) \log 3 / \log 2 \rceil$.

The structure function for a binary string will then be constant $h_x(0)$ until it hits the $n - m + 1$ curve, because the only types of automata allowed have one or two self-loops at the repeatable state. For a ternary string, there will be one more phase: Suppose as a random example $x = 1010020210$. The longest run is 00 giving $h_x(0) = n + 1 - 2 = 9$. The longest binary run is 10100 or 00202 giving $2^5 \leq 3^m$, $h_x(\lceil 5 \log 2 / \log 3 \rceil = 4) = 6$. So we get the structure function 99876654321. Now we can talk about explanatory power: this model says that the string is totally random except for one specified simple block. But what it does is identify whether a certain binary block is more surprising than another unary block, or a ternary block, which is good. In this case, having a run of two is very likely: $1 - (2/3)^{n-1} = 97.4\%$. The probability of having a binary run of five is at most

$$\frac{6 \cdot 2^5 \cdot 3^5}{3^{10}} = 0.79,$$

so the best explanation for this sequence, as having come from a distribution with a restricted alphabet block, is that there is a 2-letter-alphabet block of size 5.

If alphabet size is not restricted, we get a nontrivial notion.

Theorem 23. *We can compare probabilities of runs from different alphabet sizes in polynomial time.*

Proof. We can tabulate the cumulative distribution function effectively. Indeed, let R_n denote the longest run of heads in a sequence of n coin tosses. As mentioned by Alikhani [1] and Schilling [5], for $x \leq n - 1$,

$$\begin{aligned} \Pr(R_n \leq x) &= \sum_{i=1}^{x+1} \Pr(R_n \leq x \mid H^{i-1}T) \Pr(H^{i-1}T) \\ &= \sum_{i=1}^{x+1} \Pr(R_{n-i} \leq x) \Pr(H^{i-1}T) \\ &= \sum_{i=1}^{x+1} \Pr(R_{n-i} \leq x) p^{i-1} (1-p) \end{aligned}$$

Here H is an outcome in the restricted alphabet. So in the alphabet $\{0, 1, 2\}$ we would find the longest run from each of the following alphabets:

$$\{0\}, \{1\}, \{2\}, \{0, 1\}, \{0, 2\}, \{1, 2\}.$$

For the first three we have $p = 1/3$ and for the last three we have $p = 2/3$. \square

Example 24. *There are four nonisomorphic examples for $n = 3$: 000: unary run length 3 001: unary run length 2, binary run length 3 010: binary run length 3 012: no runs at all The most interesting is 001. Here we can use two states and get 1 accepted string, or 1 state and get 4 accepted strings. The probability of a unary run of length at least 2 is: $(1/3) + (1/3) - (1/3)(1/3) = 5/9$. The probability of a binary run of length 3 here is: $1 - (3/3)(2/3)(1/3) = 7/9$. Of course any unary run of length 2 would have to be part of a binary run of length 3, so it is not a really interesting case.*

Example 25. *For $n = 4$ we have: 0010 unary length 2, binary length 4 (interesting) (0011 is similar, with two unary length 2s) For 0010, the probability of a ternary string of length 4 being binary is $3(2/3)^4 - 3(1/3)^4 = (16 - 1)/27 = 5/9$, exactly the same as the probability of a ternary string of length 4 having a unary run of length 2.*

4.2 Algorithm for structure function of multi-run complexity for fixed valency

Next we could allow several repeat states but no going-back edges as in Figure 3 and Figure 4. It measures the presence of a collection of blocks of different

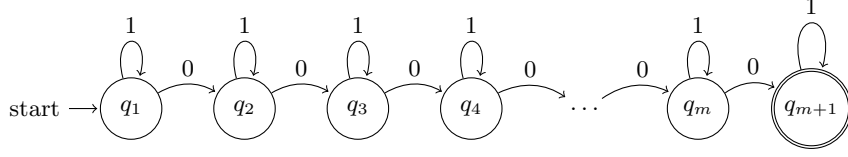


Fig. 3: An automaton illustrating multi-run complexity for a string of length n containing m many 0s, and $n - m$ many 1s.

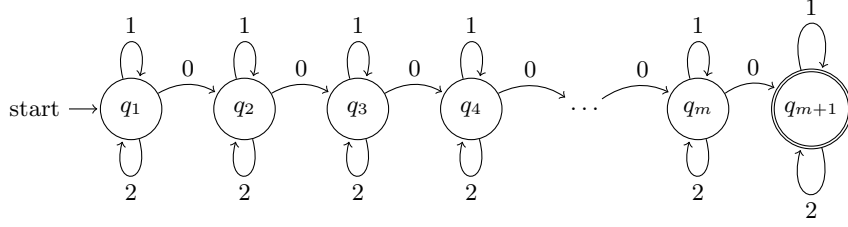


Fig. 4: An automaton illustrating multi-run complexity for a ternary string of length n containing m many 0s, and $n - m$ many 1s and 2s.

kinds; in particular it detects Bernoulli distributions and even detects changes from one Bernoulli distribution to another. It does not detect things like $(01)^*$. Considering for instance $x = 011122222$, we can imagine that it is better to use two repeat states than just one.

Theorem 26. *For multi-run complexity, there is a polynomial-time algorithm to determine whether $h_w(m) \leq q$ for a binary alphabet $\{0, 1\}$.*

Proof. We will look for automata with $\ell \leq q$ many self-loops. Let ℓ be minimal such that there is a solution to

$$x_1 + \cdots + x_\ell \geq n + 1 - q$$

consisting of lengths of disjoint runs in w . The number of solutions (x_i) , $x_i \geq 0$, of $x_1 + \cdots + x_\ell = n + 1 - q$ is

$$\binom{n + 1 - q + \ell - 1}{\ell - 1}$$

and we want to know whether

$$\binom{n + 1 - q + \ell - 1}{\ell - 1} \leq b^m$$

where $b = 2$ is the alphabet size. \square

We can extend this argument to the general case, but the polynomial-time algorithm will only work when all runs are required to have the same valence.

When the valences can vary, we cannot simply form a decreasing sequence of all the longest runs, but have to consider arbitrary collections of disjoint runs.

Theorem 27. *For multi-run complexity in an arbitrary finite alphabet of, say, size b , $h_w(m) \leq q$ iff there exist disjoint runs having lengths x_1, \dots, x_ℓ and valences v_1, \dots, v_ℓ , $\ell \leq q$, (where for instance $v_i = \{1, 2\}$ means that the run consists of 1s and 2s only; $|v_i| = 2$ is the cardinality of $\{1, 2\}$) such that $\sum x_i \geq n + 1 - q$, and*

$$\sum \left\{ \prod_{i=1}^{\ell} |v_i|^{x_i} : \sum_{i=1}^{\ell} x_i = n + 1 - q, x_i \geq 0 \right\} \leq b^m.$$

For instance, if $|v_i|$ is a constant v , this says

$$v^{n+1-q} \left| \left\{ (x_i) : \sum_{i=1}^{\ell} x_i = n + 1 - q, x_i \geq 0 \right\} \right| \leq b^m,$$

or equivalently

$$v^{n+1-q} \binom{n+1-q+\ell-1}{\ell-1} \leq b^m.$$

If $v = b$, in other words we allow no runs at all, only reducing the number of states by the cop-out of allowing arbitrary symbols, which shows $h_w(m) \leq n + 1 - m$, then we can let $\ell = 1$ and then this says $n + 1 - q \leq m$, i.e., $h_w(m) = n + 1 - m$.

5 Upper bounds on structure function for automatic complexity

Theorem 28. *The automatic structure function of a string x of length n is a function $h_x : [0, n] \rightarrow [0, \lfloor n/2 \rfloor + 1]$. Assume x is a binary string, so the alphabet size $b = 2$. Let*

$$\begin{aligned} \tilde{h} : [0, 1] &\rightarrow [0, 1/2] \\ \tilde{h}(a) &= \limsup_{n \rightarrow \infty} \max_{|x|=n} \frac{h_x(\lfloor a \cdot n \rfloor)}{n} \end{aligned}$$

where $\lfloor x \rfloor$ is the nearest integer to x . We have the following upper bound for \tilde{h} :

$$\tilde{h}(a) \leq u(a) := \begin{cases} \frac{1}{2} - \mathcal{H}^{-1}(a) & a \leq \mathcal{H}(\frac{1}{2} - \frac{\sqrt{3}}{4}) \approx 0.35 \\ \frac{2-a}{\alpha} & \mathcal{H}(\frac{1}{2} - \frac{\sqrt{3}}{4}) \leq a \leq \frac{\alpha-2}{\alpha-1} \approx 0.64 \\ 1-a & \frac{\alpha-2}{\alpha-1} \leq a \leq 1. \end{cases}$$

where

$$\alpha = \frac{4}{\sqrt{3}} \left(2 - \mathcal{H} \left(\frac{1}{2} - \frac{\sqrt{3}}{4} \right) \right) = \mathcal{H}' \left(\frac{1}{2} - \frac{\sqrt{3}}{4} \right) \approx 0.379994.$$

and where \mathcal{H} is the entropy function (Definition 12). Note that

$$u^{-1}(p) = \begin{cases} \mathcal{H}(\frac{1}{2} - p) & \frac{\sqrt{3}}{4} \leq p \leq \frac{1}{2}, \\ 2 - \alpha p & \frac{1}{\alpha-1} \leq p \leq \frac{\sqrt{3}}{4}, \\ 1 - p & 0 \leq p \leq \frac{1}{\alpha-1}. \end{cases}$$

Proof. Consider a path of length n through a Kayleigh graph with $q = pn$ many states. Let t_1 be the time spent before reaching the loop state for the first time. Let t_2 be the time spent after leaving the loop state for the last time. Let s be the number of self-loops taken by the path. Let us say that *meandering* is the process of leaving the loop state after having gone through a loop, and before again going through a loop. For fixed p let

$$\gamma(t_1, t_2, s, n) = \binom{t_1}{\frac{t_1 - pn}{2}} \binom{t_2}{\frac{t_2 - pn}{2}} \binom{n - t_1 - t_2}{s} \binom{n - t_1 - t_2 - s}{\frac{n - t_1 - t_2 - s}{2}} b^s$$

Then the number of such paths is²

$$N \leq \sum_s \sum_{t_1} \sum_{t_2} \gamma(t_1, t_2, s, n) \quad (1)$$

since half of the meandering times must be backtrack times. Since

$$\limsup_{n \rightarrow \infty} \frac{\log_b \sum_1^n a_i}{n} \leq \limsup_{n \rightarrow \infty} \frac{\log_b(n \cdot \max a_i)}{n} = \limsup_{n \rightarrow \infty} \frac{\log_b \max a_i}{n},$$

the sums can be replaced by maxima, i.e.,

$$\limsup_{n \rightarrow \infty} \frac{\log_b N}{n} \leq \limsup_{n \rightarrow \infty} \frac{\log_b \gamma(t_1, t_2, s, n)}{n}, \quad (t_1, t_2, s) \in \arg \max \gamma(\cdot, \cdot, \cdot, n).$$

By Theorem 13,

$$\limsup_{n \rightarrow \infty} \frac{\gamma(t_1, t_2, s, n)}{n} \leq \limsup_{n \rightarrow \infty} \frac{\delta(t_1, t_2, s, n)}{n}$$

where

$$\begin{aligned} \delta &= \sum_{i=1}^2 t_i \hat{\mathcal{H}} \left(\frac{1}{2} - \frac{pn}{2t_i} \right) + (n - t_1 - t_2) \hat{\mathcal{H}} \left(\frac{s}{n - t_1 - t_2} \right) + (n - t_1 - t_2 - s) \hat{\mathcal{H}} \left(\frac{1}{2} \right) + s \\ &= \sum_{i=1}^2 t_i \hat{\mathcal{H}} \left(\frac{1}{2} - \frac{pn}{2t_i} \right) + (n - t_1 - t_2) \hat{\mathcal{H}} \left(\frac{s}{n - t_1 - t_2} \right) + n - t_1 - t_2 + (1 - 1/\log_2 b)s, \end{aligned}$$

where $\hat{\mathcal{H}} = \mathcal{H}/\log_2 b$. Note that $\hat{\mathcal{H}}(1/2) = 1/\log_2 b$. Now let $\Delta(T_1, T_2, r) = \delta(T_1 n, T_2 n, r n, n)/n$ for any n . It does not matter which n , since

$$\Delta(T_1, T_2, r) = \sum_{i=1}^2 T_i \hat{\mathcal{H}} \left(\frac{1}{2} - \frac{p}{2T_i} \right) + (1 - T_1 - T_2) \hat{\mathcal{H}} \left(\frac{r}{1 - T_1 - T_2} \right) + 1 - T_1 - T_2 + (1 - 1/\log_2 b)r.$$

² We can actually replace $\binom{n - t_1 - t_2}{s}$ by $\binom{(n - t_1 - t_2 + s)/2}{s}$, since the number of non-loops between loops must be even. This would give a better upper bound, but would be harder to analyze using elementary functions.

Lemma 29. $\Delta(T_1, T_2, r)$ is maximized at $T_1 = T_2$.

Proof. Rewriting with $T = T_1 + T_2$ and $\epsilon = T_1 - T_2$, it suffices to show that with $g(x) = x\mathcal{H}(1/2 - 1/x)$, the function $f(\epsilon) = g(x + \epsilon) + g(x - \epsilon)$ is maximized at $\epsilon = 0$. This is equivalent to g being concave down, which is a routine verification. \square

In light of Lemma 29, we now let $\Delta(T, r) = \Delta(T/2, T/2, r)$, so that

$$\Delta(T, r) = T\hat{\mathcal{H}}\left(\frac{1}{2} - \frac{p}{T}\right) + (1 - T)\hat{\mathcal{H}}\left(\frac{r}{1 - T}\right) + 1 - T + (1 - \log_b 2)r.$$

Lemma 30. $\partial\Delta/\partial r = 0$ has the solution $r = (1 - T)^{\frac{b}{b+2}}$.

Proof. Note that the inverse function of the derivative $\mathcal{H}'(x) = \log_2(1-x) - \log_2 x$ is $y \mapsto \frac{1}{2^y + 1}$. Thus, we calculate

$$\frac{\partial\Delta}{\partial r} = \hat{\mathcal{H}}'(r/(1 - T)) + 1 - 1/\log_2 b = 0$$

$$\mathcal{H}'(r/(1 - T)) + \log_2 b - 1 = 0$$

$$\frac{r}{1 - T} = (\mathcal{H}')^{-1}(1 - \log_2 b) = \frac{1}{2^{1 - \log_2 b} + 1} = \frac{1}{\frac{2}{b} + 1} = \frac{b}{b + 2}$$

\square

Then

$$\begin{aligned} \lim_{n \rightarrow \infty} \frac{\log_2 N}{n} &\leq \varphi(T, p) := \Delta(T, (1 - T)^{\frac{b}{b+2}}) = \\ &= T\hat{\mathcal{H}}\left(\frac{1}{2} - \frac{p}{T}\right) + (1 - T)c_b \end{aligned}$$

where

$$c_b := \left(1 + \hat{\mathcal{H}}\left(\frac{b}{b+2}\right) + (1 - 1/\log_2 b)\frac{b}{b+2}\right)$$

Note that $c_2 = 2$.

Lemma 31. Fix $0 \leq p \leq 1/2$ and assume $2p \leq T \leq 1$. Then we have $0 < \frac{\partial\varphi}{\partial T}$ iff

$$T < T(p) := \frac{p}{\sqrt{\frac{1}{4} - \left(\frac{2}{b(2+b)}\right)^2}} = \frac{2p}{\sqrt{1 - \left(\frac{4}{b(2+b)}\right)^2}} = \frac{4p}{\sqrt{4 - \left(\frac{8}{b(2+b)}\right)^2}} = \frac{4p}{\sqrt{3}}, \quad b = 2.$$

Let

$$L_b = \frac{2p}{T(p)} = \sqrt{1 - \left(\frac{4}{b(2+b)}\right)^2} \leq 1.$$

Note that $T(p) \leq 1$ iff $p \leq L_b/2$, and

$$\begin{aligned} \varphi(T(p), p) &= T(p)\hat{\mathcal{H}}\left(\frac{1}{2} - \frac{p}{T(p)}\right) + (1 - T(p))c_b \\ &= \frac{p}{L_b}\hat{\mathcal{H}}\left(\frac{1}{2} - L_b\right) + (1 - \frac{p}{L_b})c_b \\ &= c_b - \left(c_b - \hat{\mathcal{H}}\left(\frac{1}{2} - L_b\right)\right)\frac{p}{L_b} =: c_b - \alpha_b p \end{aligned}$$

Proof. Let $\beta(T) = \frac{1}{2} - \frac{p}{T}$. We have

$$\begin{aligned} \frac{\partial \varphi}{\partial T} &= \hat{\mathcal{H}}\left(\frac{1}{2} - \frac{p}{T}\right) + T\hat{\mathcal{H}}'\left(\frac{1}{2} - \frac{p}{T}\right)\left(\frac{p}{T^2}\right) - c_b = \hat{\mathcal{H}}(\beta) + T\hat{\mathcal{H}}'(\beta)\left(\frac{p}{T^2}\right) - c_b \\ &= -\beta \log_b \beta - (1 - \beta) \log_b(1 - \beta) + (\log_b(1 - \beta) - \log_b \beta)(p/T) - c_b \\ &= -\beta \log_b \beta - (1 - \beta) \log_b(1 - \beta) + (\log_b(1 - \beta) - \log_b \beta)(1/2 - \beta) - c_b \end{aligned}$$

Now $b^0 < b^{\partial \varphi / \partial T}$ iff

$$\begin{aligned} 1 &< \beta^{-\beta}(1 - \beta)^{-(1-\beta)}((1 - \beta)/\beta)^{1/2-\beta}b^{-c_b} \\ 1 &> \beta^\beta(1 - \beta)^{(1-\beta)}\left(\frac{\beta}{1 - \beta}\right)^{1/2-\beta}b^{c_b} = \beta^{1/2}(1 - \beta)^{1/2}b^{c_b} \\ 1 &> \beta(1 - \beta)b^{2c_b}, \quad 0 \leq \beta \leq 1/2 \end{aligned}$$

giving

$$\begin{aligned} \beta &< \frac{1 - \sqrt{1 - 4b^{-2c_b}}}{2} = \frac{1 - \sqrt{3/4}}{2}, \quad b = 2 \\ p/T &= 1/2 - \beta > \frac{\sqrt{1 - 4b^{-2c_b}}}{2} = \frac{\sqrt{3/4}}{2}, \quad b = 2 \\ T &< \frac{2}{\sqrt{1 - 4b^{-2c_b}}}p = \frac{4}{\sqrt{3}}p, \quad b = 2 \end{aligned}$$

Note that $b \geq 2$ and $4b^{-2c_b} = \frac{2^4}{b^2(b+2)^2} = \left(\frac{4}{b(b+2)}\right)^2$ give $1 - 4b^{-2c_b} > 0$, as required. \square

Hence

$$\begin{aligned} \lim_{n \rightarrow \infty} \frac{\log_2 N}{n} &\leq \psi(p) := \varphi(\min\{1, T(p)\}, p) \\ &= \begin{cases} \varphi(1, p) = \hat{\mathcal{H}}(1/2 - p), & p \geq L_b/2; \\ \varphi(T(p), p) = c_b - \alpha_b p, & p \leq L_b/2. \end{cases} \end{aligned}$$

Consequently $\tilde{h}(\psi(p)) \leq p$. Note that $\psi = u^{-1}$. \square

As Theorem 28 shows, the largest number of paths is obtained by going *fairly* straight to the loop state; spending half the time looping and half the time meandering; and then finally going equally fairly straight to the start state. The optimal value of r obtained shows that half of the time between first reaching the loop state and finally leaving the loop state should be spent looping.

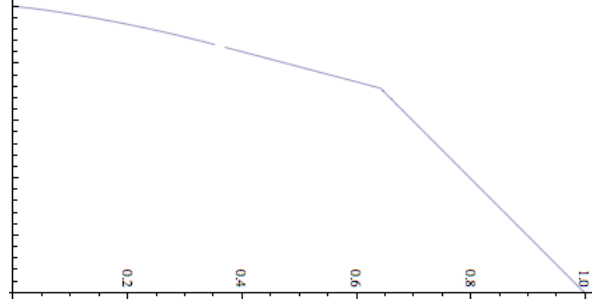


Fig. 5: Bounds for the automatic structure function for alphabet size $b = 2$; see Theorem 28.

References

1. Malihe Alikhani. American option pricing and optimal stopping for success runs. Master's thesis, University of Hawaii at Manoa, U.S.A., 2013.
2. Bjørn Kjos-Hanssen. Structure Function for Run Complexity. <http://kjos-hanssen.appspot.com/structure-function>, July 2014.
3. Kayleigh Hyde. Nondeterministic finite state complexity. Master's thesis, University of Hawaii at Manoa, U.S.A., 2013.
4. Bjørn Kjos-Hanssen and Kayleigh Hyde. Nondeterministic automatic complexity of almost square-free and strongly cube-free words. In *COCOON 2014*, volume 8591 of *Lecture Notes in Comput. Sci.*, pages 61–70. Springer, Heidelberg, 2014.
5. Mark F. Schilling. The longest run of heads. *College Math. J.*, 21(3):196–207, 1990.
6. Jeffrey Shallit and Ming-Wei Wang. Automatic complexity of strings. *J. Autom. Lang. Comb.*, 6(4):537–554, 2001. 2nd Workshop on Descriptive Complexity of Automata, Grammars and Related Structures (London, ON, 2000).
7. Ludwig Staiger. The Kolmogorov complexity of infinite words. *Theoret. Comput. Sci.*, 383(2-3):187–199, 2007.
8. Nikolai K. Vereshchagin and Paul M. B. Vitányi. Kolmogorov's structure functions and model selection. *IEEE Trans. Inform. Theory*, 50(12):3265–3290, 2004.